Title	:	HIGH-LEVEL PROGRAM INTERFACE FOR GRAPHICS OPERATIONS		
Inventors	:	JOHN HARPER et al		
Serial No.	:	10/826,762	Filed	: April 16, 2004
Examiner	:	Joni Hsu	Art Unit	: 2628
Docket	:	119-0033US	Customer	: 61947

Mail Stop Appeal Brief Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

This Appeal Brief is filed in response to the Final Office Action mailed on July 21, 2008.

I. REAL PARTY IN INTEREST

Apple Inc. is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

None.

III. STATUS OF CLAIMS

Claims 1-85 are rejected. Claims 1-85 are appealed.

IV. STATUS OF AMENDMENTS

None.

V. SUMMARY OF CLAIMED SUBJECT MATTER

This section provides a concise explanation of the subject matter defined in each independent claim involved in this appeal, referring to the specification by paragraph and line number and to the drawings by reference characters as required by 37 C.F.R. 41.37(c)(I)(v). It should be noted, citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element. All references to page and line numbers below are per the specification as filed on 21 May 2002 and amendments to claims filed on 14 Feb 2006. Additionally, references are not necessarily exhaustive, and various claim elements may also be described at other locations.

Generally, Assignee claims methods (independent claims 1, 28, 43, 58, 63, 74, and 75), a system (independent claim 22), an application program interface (independent claims 79 and 84) and a computer-readable medium (claim 85) to provide a high-level program interface for graphics operations.

Independent claim 1 recites a method of editing an initial image (¶ 43, 8. 1-3), the method comprising the steps of:

- a first process requesting a filter (¶ 44, 9. 1-7) from a second process;
- said first process defining a relationship (¶ 51, 10. 8-10) between said filter (¶ 44, 9. 1-7) and said initial image (¶ 43, 8. 1-3), said related filter (¶ 44, 9. 1-7) and initial image (¶ 43, 8. 1-3) comprising a program;
- said second process compiling said program, yielding a compiled program;
- running at least a portion of said compiled program to apply a function of said filter (¶ 44, 9. 1-7) to said image (¶ 43, 8. 1-3), yielding an pixel-image (¶ 70, 17. 15-16) result.

Independent claim 22 recites a system for editing an initial image (¶ 43, 8. 1-3), comprising:

- a first microprocessor (Fig. 1, element 11) running a first process and a second process for servicing requests from said first process;
- a memory (Fig. 1, element 114) for storing a filter (¶ 44, 9. 1-7), said filter (¶ 44, 9. 1-7) requested by said first process;
- a second memory (Fig. 1, element 114) for storing a data structure (¶ 51, 10. 5-7) comprising a relationship (¶ 51, 10. 8-10) between said initial image (¶ 43, 8. 1-3) and said filter (¶ 44, 9. 1-7), said first process causing the creation of said data structure (¶ 51, 10. 5-7);
- a second microprocessor (Fig. 1, element 112) for running a program created using said data structure (¶ 51, 10. 5-7);
- a third memory (Fig. 1, element 114 or element 113) for storing a pixel image resulting (¶ 70, 17. 15-16) from running said program.

Independent claim 28 recites a method of creating a result image (¶ 70, 17. 15-16) comprising the steps of:

- a first process requesting the creation of a context (¶ 45, 9. 1-3);
- said first process requesting the creation of a result image (¶ 70, 17. 15-16);
- said first process indicating parameters associated with the creation of said result image (¶ 70, 17. 15-16);
- said first process requesting that said result image (¶ 70, 17. 15-16) be rendered to said context (¶ 45, 9. 1-3);

- a second process servicing said requests of said first process, said servicing comprising the steps of:
    - optimizing a graph (¶ 70, 17. 3-13; Fig. 4) representing said result image (¶ 70, 17. 15-16);
    - compiling said optimized graph (¶ 70, 17. 12-15);
    - causing the rendering of said compiled optimized graph into said context (¶ 70, 17. 15-16).

Independent claim 43 recites a method of creating a result image (¶ 70, 17. 15-16) comprising the steps of:

- a first process running on a first microprocessor (Fig. 1, element 11) requesting the creation of a context (¶ 55, 11. 1-4);
- said first process (¶ 34, 6. 1-12) requesting the creation of a result image (¶ 70, 17. 15-16);
- said first process indicating parameters (¶ 53, 11. 1-2; Fig. 5, examples) associated with the creation of said result image (¶ 70, 17. 15-16);
- said first process requesting that said result image (¶ 70, 17. 15-16) be rendered to said context (¶ 45, 9. 1-3);
- a second process running on said first microprocessor (Fig. 1, element 11) servicing said requests of said first process, said servicing comprising the steps of:
    - optimizing a graph (¶ 36, 7. 1-11; Fig. 4) representing said result image (¶ 70, 17. 15-16);
    - compiling said optimized graph (¶ 53, 11. 1-12);

- causing rendering of said compiled optimized graph into said context (¶ 53, 11. 1-12), said rendering occurring on a second microprocessor (Fig. 1, element 112).

Independent claim 58 recites a method for providing a high level interface to a graphics processing resource comprising:

- a first process requesting performance of a task through one or more function calls serviced by said graphics processing resource (¶ 33-34, 6.; Fig. 3, element C101), said function calls selected from one or more of the following options, (i) creating of an image (¶ 57, 12. 1-7), (ii) creating a filter (¶¶ 59-60, 13. all), (iii) setting arguments associated with said filter (¶ 65, 16. 1-5), (iv) asking for the output of said filter (¶ 67, 16. 1-6) or another filter, (v) creating a context (¶ 55, 11. 1-5); and (vi) rendering an image (¶ 70, 17. 1-3) to said context or another context (¶ 45, 9. 1-3);

- said request having an object associated therewith, said object comprising one of the following: a context (¶ 45, 9. 1-3), an image (¶ 43, 8. 1-3), a filter (¶ 44, 9. 1-7), or a vector (¶ 46, 9. 1-6);

- said request defining a relationship (¶ 51, 10. 8-10) between at least one of said functions and one of said objects;

- said graphics processing resource (¶ 33-34, 6.; Fig. 3, element C101) servicing said request.

Independent claim 63 recites a method for providing a high level interface to a graphics processing resource comprising:

- a first process requesting performance of a task through one or more function calls serviced by said graphics processing resource (¶ 33-34, 6.; Fig. 3, element C101), said function calls selected from one or more of the following options, (i) creating of an image (¶ 57, 12. 1-7), (ii) creating a context (¶ 55-56, 11-12. all); and (v) rendering

an image (¶ 43, 8. 1-3) to said context (¶ 70, 17. 1-16) or another context (¶ 45, 9. 1-3);

- said request having an object associated therewith, said object comprising one of the following: a context (¶ 45, 9. 1-3), or an image (¶ 43, 8. 1-3);

- said request defining a relationship (¶ 51, 10. 8-10) between at least one of said functions and one of said object;

- said graphics processing resource (¶ 33-34, 6.; Fig. 3, element C101) servicing said request.

Independent claim 74 recites a method for rendering an image (¶ 70-71, 17. all) comprising the steps of:

- a first process running on a CPU (¶ 9, 3. 1-12; Fig. 1, element 11) requesting the creation of an image (¶ 57-58, 12. all);

- a graphics services resource (¶ 33-34, 6.; Fig. 3, element C101), responding to said request by running a first routine on said CPU (¶ 9, 3. 1-12; Fig. 1, element 11), said first routine for optimizing a graph (¶ 70, 17. 3-15; Fig. 4) representing said image (¶ 43, 8. 1-3);

- said first process requesting the rendering of said image (¶ 70-71, 17. all) to a specified context (¶ 45, 9. 1-3);

- said graphics services resource (¶ 33-34, 6.; Fig. 3, element C101) causing the rendering of said graph (¶ 70, 17. 15-16; Fig. 4) representing said image (¶ 43, 8. 1-3), said rendering occurring on a GPU (¶ 9, 3. 1-12; Fig. 1, element 112).

Independent claim 75 recites a method for converting a first image (¶ 43, 8. 1-3) representation into a second image (¶ 43, 8. 1-3) representation, comprising the steps of:

- creating a first graph (¶ 36, 7. 1-11; Fig. 4) associated with said first image (¶ 43, 8. 1-3) representation where software routines for creating such graph (¶ 36, 7. 1-11; Fig. 4) execute on a CPU (¶ 9, 3. 1-12; Fig. 1, element 11);

- determining an intersection of the first graph's global domain of definition (¶ 92, 24. 1-8) and global region of interest (¶ 98, 26. 1-11);

- resolving a first node in said first graph by running software routines on said CPU (¶ 9, 3. 1-12; Fig. 1, element 11) to (i) determine if said first node may be combined with a second node and (ii) create program steps for calculating and storing only the portions of any intermediary image (¶ 98, 26. 8-14) that relate to said intersection (¶ 101, 27. 8-10), said program steps for compilation on said CPU (¶ 9, 3. 1-12; Fig. 1, element 11) and execution on a GPU (¶ 9, 3. 1-12; Fig. 1, element 112).

Independent claim 79 recites an image (¶ 43, 8. 1-3) processing application program interface embodied on one or more computer readable media, comprising: a first group of services related to filter objects (¶ 59, 13. 1-9); a second group of services related to image objects (¶ 57-58, 12. all); a third group of services related to context objects (¶ 55, 11. 1-5); and a fourth group of services related to vector objects (¶ 65, 16. 3-5).

Independent claim 84 recites an application program interface for facilitating image (¶ 43, 8. 1-3) processing, the application program interface comprising functions to: create image objects (¶ 57-58, 12. all); create context objects (¶ 55-56, 11-12. all); create filter objects (¶ 59, 13. 1-9; Fig. 5 , element 52); set filter object parameters (¶ 65, 16. 1-9); obtain filter object output (¶67, 16. 1-4); and convert image objects into context objects (¶ 70, 17. 1-16).

Independent and multiple dependent claim 85 recites a computer-readable medium having computer executable instructions for performing the method recited in any one of claims 1, 28, 43, 58, 63, 74 or 75.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 79-85 stand rejected under 35 U.S.C § 101 as allegedly being directed to non-statutory subject matter.

Claims 28-29, 38-39, 43-44, 53-54 and 74 stand rejected under 35 U.S.C. § 102(b) as allegedly being anticipated by U.S. Patent No. 6,215,495 to Grantham ("Grantham").

Claims 75 and 78 stand rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent No. 6,995,765 to Boudier ("Boudier").

Claims 78-84 stand rejected under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent No. 6,600,840 to McCrossin ("McCrossin").

Claims 1-2, 8, 11-12, 14-20, 36-37, 40-41, 51-52, 55-56, 58, 61-63, 66-73 and 85 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham in view of McCrossin.

Claim 3 stands rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham in view of McCrossin further in view of U.S. Publication 2002/0033844 to Levy ("Levy").

Claims 4-6 and 21 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham and McCrossin further in view of Boudier.

Claims 7, 9, 60 and 65 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham and McCrossin further in view of U.S. Patent No. 6,411,301 to Parikh ("Parikh").

Claim 10 stands rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham, McCrossin and Parikh further in view of U.S. Patent No. 6,867,779 to Doyle ("Doyle").

Claim 13 stands rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham and McCrossin further in view of U.S. Patent No. 6,801,202 to Nelson ("Nelson").

Claims 22-23 and 25-27 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham and McCrossin further in view of U.S. Patent No. 6,919,906 to Hoppe ("Hoppe").

Claim 24 stands rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham, McCrossin and Hoppe further in view of U.S. Patent No. 5,854,637 to Sturges ("Sturges").

Claims 30-32 and 45-47 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham and Boudier.

Claims 33 and 48 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham and Boudier further in view of U.S. Publication 2002/0033844 to Levy ("Levy").

Claims 34-35 and 49-50 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham in view of Levy.

Claims 42 and 57 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham in view of U.S. Patent No. 6,977,661 to Stokes ("Stokes").

Claims 59 and 64 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham and McCrossin in view of U.S. Patent No. 6,877,779 to Doyle ("Doyle").

Claims 76-77 stand rejected under 35 U.S.C. § 103(a) as allegedly being obvious over Boudier in view of Doyle.

VII. ARGUMENT

All the claims do not stand or fall together. Instead, Assignee presents separate arguments for various groups of independent and dependent claims. Any claim argued separately is under a subheading identifying the claim by number. Claims argued as a group are under a subheading identifying the claims by number. After a concise discussion of the cited art, each of these arguments is separately presented below under separate headings and sub-headings as required by 37 C.F.R. 41.37(c)(l)(vii).

A. Section 101 Rejections

Claims 79-85

The Examiner has rejected claims 79-85 under 35 U.S.C. §101 as being allegedly being directed to non-statutory subject matter. Specifically, the Examiner has concluded that one of ordinary skill in the art would understand that "computer readable media" could include signals to maintain his § 101 rejection.

Regarding claims 79-84 and 85, the Examiner asserts the specification does not appear to explicitly define "computer-readable medium" and that one of ordinary skill in the art would fairly determine this to include signals or other forms of propagation and transmission media. *See* Final Office Action dated 21 July 2008 at pg. 2. The Board is directed to the original specification at least at pg. 5 ¶ 32. This portion of the specification states "memory or any other suitable storage medium that exists or may exist in the future." Furthermore, specific examples (*e.g.* RAM, computer disk drive, non-volatile storage, optical memory, USB devices) of storage are listed here and throughout the specification. Assignee points out that one of ordinary skill in the art will recognize that a computer readable medium is clearly a storage medium and clearly defined in the specification.

Additionally, the Examiner asserts that claim 84 pertains to non-statutory subject matter because it is "descriptive material *per se*." To explain this rejection the Examiner presents differing definitions for "data structures" and "APIs" but contradicts those definitions by incorrectly concluding that "*one of ordinary skill in the art would* ***consider API to be a data structure.***" Final Office Action dated 21 July 2008 at p. 9 ¶ 20.

Furthermore, each of these claims includes a concrete and tangible step of processing an image and cannot be classified as "nonfunctional." In particular, among other limitations, the claims recite functions to "create image objects" and "convert image objects into context objects," which are limitations that perform a defined function. Furthermore, "[o]nly when the claimed invention taken as a whole is directed to a mere program listing *i.e.*, to only its description or expression, is it descriptive material *per se* and hence nonstatutory.... When a computer program is claimed in a process where the computer is executing the computer program's instructions, USPTO personnel should treat the claim as a process claim." MPEP § 2106.01. Therefore the requirements of 35 U.S.C. § 101 are clearly met. Assignee respectfully requests the Board withdraw this rejection.

Legal Principles Regarding Prior Art Rejections

Even though the Examiner is entitled to give the claims their broadest reasonable interpretation (*See* M.P.E.P. 2111), the Examiner cannot read the claim language so broadly as to completely ignore the explicit language actually recited in the claims. In fact, each word in Assignee's claim must be considered in determining whether the claims are patentable over the cited prior art. *See* M.P.E.P. 2143.03. In addition, every element of Assignee's claims must be identically shown in the cited prior art for the cited prior art to anticipate in terms of 35 U.S.C. 102. *See* Diversitech Corp. v. Century Steps, Inc., 850 F.2d 675, 677, 7 U.S.P.Q.2d 1315, 1317 (Fed. Cir. 1988); *See* also Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 U.S.P.Q.2d 1913, 1920 (Fed. Cir. 1989), cert. denied, 493 U.S. 853 (1989) (The "identical invention must be shown in as complete detail as is contained in the patent claim"). Therefore, as will be explained below, the cited references of Grantham, Boudier and McCrossin do not disclose each element contained in the allegedly anticipated claims, and there is substantial difference between the claimed invention and the disclosures of Grantham, Boudier and the other cited art. Furthermore, the attempt to combine these references with the cited secondary references also fails to disclose each and every claimed element and results in awkward and non-obvious combinations. For each of the Examiner's rejections below, Assignee will show that the Examiner has largely taken the references out of context or improperly extrapolated from those references and then proceeded to combine them in an unreasoned manner. Therefore, the

Examiner has failed to establish a *prima facie* case of either anticipation or obviousness required to sustain these rejections.

B. Section 102 Rejections

The Examiner has rejected claims 28-29, 38-39, 43-44, 53-54 and 74 under 35 U.S.C. § 102(b) as allegedly being anticipated by U.S. Patent No. 6,215,495 to Grantham ("Grantham").

Grantham

Grantham is directed to "a platform independent application program interface for interactive three-dimensional scene management." Grantham at Abstract. Grantham further discloses "[i]n particular, the applications programming interface is used to render a three-dimensional scene according to a download file. This is accomplished by building a scene graph from a number of different objects which are stored in local memory [as opposed to a server on the internet]. These objects have variables which can be changed by subroutine calls." Grantham at Col. 2, lns. 54-59. Grantham discloses a use for Virtual Reality Meta Language (VRML) in recognition of the limited bandwidth of the internet to more efficiently utilize the existing bandwidth in the transmission of 3-D image information. *See* Grantham at Col. 1, lns. 64-67.

To summarize, Grantham discloses a use for the ***interpreted*** VRML language that stores persistent data objects in the computer's local memory such that the server, which is located at some point in the internet, is saved from having to re-transmit this type of data each and every time a new scene graph is to be constructed. *See* Grantham at Col. 4, lns. 50-55. Of particular note, Grantham repeatedly states that VRML is an interpreted language (*See eg.* Col. 5 lns. 5, 8; Fig. 1 element 111) and that the "CompileAction class 403 compiles a specified subgraph into a ***data structure which is more efficient for traversals***." Grantham at Col. 7, lns. 50-52 (emphasis added). The "CompileAction" is NOT "compilation" in its normal defined sense or as used by assignee, because Grantham's "CompileAction" does NOT yield executables.

Claim 28

The Examiner has specifically rejected independent claim 28 as follows:

24. As per Claim 28, Grantham teaches processor 108 initiates request which is routed to server 101. When server 101 receives such request, processor 102 retrieves appropriate VRML file. VRML file that is requested instructs API 112 to make a number of function calls to various graphics engines 113 for performing desired functions on persistent data objects 118. API is structured as collection of class hierarchies (c. 4, ll. 63-c. 5, ll. 23), including Graphics State classes that includes Context, Appearance, Material, Texture, TexTransform classes (c. 28, ll. 14-16). Context class maintains graphics state for particular graphics context (c. 3, ll. 2-4). The other Graphics State classes define how result image is to be created (c. 8, ll. 38-46). So Graphics State classes indicate parameters associated with creation of result image. DrawAction class is used to draw scene (c. 7, ll. 31-33). Since VRML file that is requested includes this information about Context class, Graphics State classes, DrawAction class, VRML file that is requested requests creation of context (Context class), requests creation of result image (DrawAction class), and includes information that indicates parameters associated with creation of result image (Graphics State classes). Since 1st process requests this VRML file, 1st process is considered to request creation of context, request creation of result image, and indicate parameters associated with creation of result image. So Grantham teaches method of creating result image having 1st process requesting creation of context; 1st process requesting creation of result image; 1st process indicating parameters associated with creation of result image; 1st process requesting result image be rendered to context (c. 4, ll. 63-c. 5, ll. 23; c. 28, ll. 14-16; c. 3, ll. 2-24; c. 8, ll. 38-46; c. 7, ll. 31-33); 2nd process servicing requests of 1st process, servicing comprising steps of optimizing graph representing result image; compiling optimized graph; causing rendering of compiled optimized graph into context (c. 5, ll. 2-23; c. 4, ll. 55-58; c. 7, ll. 50-57; c. 3, ll. 2-4). Interpreter 111 modifies scene graph to suit specific graphics subsystem hardware. Interpreter enables VRML file to be adapted to run on virtually any type of machine (c. 5, ll. 5-9). So interpreter translates VRML file to another computer language that is able to be run on graphics subsystem hardware, so interpreter is considered to perform compiling.

Final Office Action dated July 21, 2008 at pp. 10-11.

As stated above, Grantham discloses communication between an internet server and a local processor's web browser. The Examiner interprets Grantham such that the 1st process is run on the local machine 108 and the second process is performed by the internet server 101. The Examiner then asserts "1st process indicating parameters associated with creation of result image." This assertion is incorrect because Grantham discloses that an internet request is initiated by the local machine and "[w]hen server 101 receives such a request, processor 102 retrieves the appropriate VRML file from memory 104. The VRML file is then transmitted back

over the Internet ... The VRML file instructs the API 112 to make a number of function calls." Grantham at Col. 4, ln 66 through Col. 5., ln 3. Therefore, it is clear that the local processor (*i.e.*, 1st process) does not "indicate parameters associated with creation of result image" as recited in the claim. The local processor is merely making an internet request of a web server and the web server provides the appropriate VRML file. The cited "first process" in Grantham ***merely asks for the VMRL file*** without any reference to or cognizance of the claimed "creation of context", "creation of result image", or "parameters associated with result image". It is simply improper for the Examiner to conclude "Since 1st process requests this VRML file, 1st process is considered to request creation of context, request creation of result image, and indicate parameters associated with creation of result image." The Examiner has apparently added many features to a simple file request on his own.

Furthermore, Grantham does not disclose the steps of both optimizing the graph and compiling the optimized graph as recited in claim 28. The Examiner cites to the CompileAction class disclosed in Grantham to anticipate this limitation. As stated above, VRML is an ***interpreted*** language and the CompileAction class creates a ***data structure*** which is more efficient for traversals. Even though Grantham has named the example class "CompileAction," the function of the class is to optimize a data structure for traversals and not the standard use of compiling which produces an executable. *See* Specification at ¶ 70. As stated above, Grantham's "CompileAction" does NOT produce an executable and thus, is not "compiling".

Because Grantham does not disclose each and every limitation of claim 28 the Examiner has failed to present a legitimate *prima facie* case of anticipation. Furthermore, each of claims 29 and 38-39 depend from claim 28 and are not anticipated by Grantham for at least the same reasons. Assignee respectfully requests the Board withdraw this rejection.

Claim 43

The Examiner has specifically rejected independent claim 43 as follows:

> 28. As per Claim 43, it is similar in scope to Claim 28, except it has additional limitation that 1st process and 2nd process are running on first microprocessor, and rendering occurs on second microprocessor. Grantham teaches first process (requesting) and second process (compiling) are running on first microprocessor (108), and rendering occurs on second microprocessor (109) (c. 4, ll. 63-c. 5, ll. 23; c. 7, ll. 50-57). So Claim 43 is rejected under same rationale as Claim 28.

Final Office Action dated 21 July, 2008 at p. 12.

The Examiner asserts that the first process is the requesting process disclosed in Grantham. As stated above, the requesting process in Grantham is a web browser and the web browser does not perform the limitations of "requesting the creation of a result image," "indicating parameters associated with the creation of said result image," and "requesting that said result image be rendered to said context," as recited in claim 43. The internet server of Grantham performs the function of creating the VRML file. The arguments stated above with respect to claim 28 also apply to claim 43. No "rendering" of a "compiled" & "optimized" "graph" is disclosed in the cited art.

Because Grantham does not disclose each and every limitation of claim 43, the Examiner has failed to present a legitimate *prima facie* case of anticipation. Furthermore, each of claims 44 and 53-54 depend from claim 43 and are not anticipated by Grantham for at least the same reasons. Assignee respectfully requests the Board withdraw this rejection.

Claim 74

The Examiner has specifically rejected independent claim 74 as follows:

> 30. As per Claim 74, Grantham teaches rendering image, 1st process running on CPU 107 requesting creation of image; graphics services resource 109, responding to request by running 1st routine on CPU, 1st routine for optimizing graph representing image; 1st process requesting rendering of image to specified context; graphics services resource causing rendering of graph representing image, rendering occurring on GPU (c. 4, ll. 55-58, 63-67; c. 5, ll. 1-23, 61-67). Grantham recites "processor 108 of computer system 116 initiating a request which is routed by

> input/output (I/O) device 107 through the Internet 106 to server 101" (c. 4, ll. 63-66). So first process is running on CPU 116 requesting creation of image. Grantham teaches request results in VRML file being transmitted to computer system 116, and VRML file instructs API 112, and API is structured as collection of class hierarchies (c. 4, ll. 63-c. 5, ll. 23), including CompileAction class 403 that compiles specified subgraph into data structure which is more efficient for traversals (c. 7, ll. 50-57). So, ultimately API 112 responds to request by running first routine on CPU 116, first routine for optimizing graph representing the image (c. 4, ll. 63-c. 5, ll. 23; c. 7, ll. 50-57), and so API 112 is considered to be graphics services resource.

Final Office Action dated 21 July, 2008 at pp. 12-13.

The 1st process of Grantham is a web browser and is not "requesting creation of an image," as recited in claim 74. The web browser is "initiating a request which is routed by input/output (I/O) device 107 through the internet 106 to server 101." Grantham at Col. 4, lns. 63-67. A web browser requesting information from a server is not the same as "a first process running on a CPU requesting creation of an image" as recited in claim 74. Furthermore, neither the web browser nor Grantham's server has "a graphics services resource, responding to said request" (Grantham's internet server 101 responds to the "request" disclosed in Grantham).

Because Grantham does not disclose each and every limitation of claim 74 the Examiner has failed to present a legitimate *prima facie* case of anticipation. Assignee respectfully requests the Board withdraw this rejection.

Claims 75 and 78

The Examiner has rejected claims 75 and 78 under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent No. 6,995,765 to Boudier ("Boudier").

Boudier

Boudier is directed to optimization of a scene graph via an optimization base that contains a set of specific atomic optimizations; an optimization registry that lists each atomic optimization; parameters associated with each optimization; and priority information relating to the necessary order in which optimizations must be performed. Boudier at Abstract. "The selection of a specific atomic optimization is made by user 410. User 410 supplies user configuration information 415 to optimization manager 440 via a configuration manager 420."

Boudier at Col. 4, lns. 49-54. To summarize, Boudier discloses a system that takes ***user input*** to perform specific ***atomic*** optimization steps for a particular scene graph.

Claim 75

The Examiner has rejected independent claim 75 specifically as follows:

> 33. As per Claim 75, Boudier teaches converting 1st image representation into 2nd image representation, comprising creating 1st graph associated with 1st image representation (c. 1, ll. 29-36; c. 2, ll. 31-34; c. 3, ll. 61-63) where software routines for creating such graph execute on CPU (504) (c. 1, ll. 49-51; c. 6, ll. 34-38), determining intersection of 1st graph's global domain of definition (input scene graph) and global region of interest (bounding box) (c. 9, ll. 12-21); resolving 1st node in 1st graphic by running software routines on CPU to determine if 1st node may be combined with 2nd node (c. 9, ll. 40-53; c. 5, ll. 33-37) and create program steps for calculating and storing only portions of any intermediary image that relate to intersection (c. 9, ll. 12-21), program steps for compilation on CPU, execution on GPU (c. 5, ll. 33-37; c. 6, ll. 19-21).

Final Office Action dated 21 July, 2008 at p. 13.

The Examiner asserts that the Assignee's graph's global domain of definition is anticipated by Boudier's input scene graph. This is simply incorrect. Assignee directs the Board to the instant specification at ¶ 92 which states "[a] practical way to think about domain of definition is as a representation of all places in an image that are explicitly defined and not transparent." The scene graph of Boudier is a model with the nodes representing features of a scene and edges representing associations between the connected components. *See* Boudier at Col. 1, lns. 30-34.

The Examiner similarly asserts that Assignee's global region of interest is equivalent to the bounding box of Boudier. This assertion is also incorrect. A bounding box is defined as the "a cupoid, or in 2-D a rectangle, containing the object." *See* http://en.wikipedia.org/wiki/Bounding_volume. That is the cuboid formed when rectangles are used to outline an objects left, right, front, back, top and bottom. This is in stark contrast to the definition of global region of interest (ROI) defined in the instant specification at ¶¶ 98 and 100 which explain that a region of interest "is the portion of the input image that is necessary to compute the given DOD."

Additionally, in his response to previous arguments, the Examiner states that DOD and ROI "are not recited in the rejected claim(s)." Final Office Action dated 21 July, 2008 at pg. 6

first full ¶. However, as can be clearly seen claim 75 does in fact include a limitation of "determining an intersection of the first graph's global domain of definition and global region of interest." Because Boudier does not disclose either a DOD or ROI as defined by Assignee, it is impossible for Boudier to anticipate a claim which, *inter alia,* determines an intersection between these two elements.

Boudier is silent regarding the region of interest or domain of definition as used in claim 75. Because Boudier does not disclose each and every limitation of claim 75, the Examiner has failed to present a legitimate *prima facie* case of anticipation. Furthermore, claim 78 depends from claim 75 and is not anticipated by Boudier for at least the same reasons. Assignee respectfully requests the Board withdraw this rejection.

Claims 79-84

The Examiner has rejected claims 79-84 under 35 U.S.C. § 102(e) as allegedly being anticipated by U.S. Patent No. 6,600,840 to McCrossin et al. ("McCrossin").

McCrossin

McCrossin is directed to a system for changing the format of an image by applying a series of filters from a filter stack. The filters are applied to the source image for each parameter of the source image that does not match a parameter of the requested or desired result image. In essence, McCrossin proposes a system to simplify the conversion of the same image from one format to another; for example, from a BMP file to a TIFF file. *See* Col. 6 line 40 *et. Seq.*

Claim 79

The Examiner has rejected independent claim 79 specifically as follows:

> 37. As per Claim 79, McCrossin teaches application calls transformation object using API call (c. 7, ll. 33-35). Transform object 103 determines actual image vector 139, which describes

format of image to be read or written (c. 6, ll. 51-53). Since this describes format of image, this is related to image objects and context objects. Image request vector is received from application and is compared to actual image vector 139 to determine what filters are needed (c. 6, ll. 53-56). Filters are accessed by transformation object from filter library. Filter library contains number of different filters available for performing various image transformations (c. 6, ll. 59-62). Pointer to filter vector in filter object 150 points to filter vector 154, which in depicted example is crop filter (c. 7, ll. 19-21). So, McCrossin teaches image processing API and API services related to filter objects, API services related to image objects, API services related to context objects and API services related to vector objects. McCrossin uses API to call at filters. So, McCrossin teaches image processing application program interface embodied on one or more computer readable media (c. 6, ll. 25-30), having 1st group of services related to filter objects (c. 6, ll. 59-67); 2nd group of service related to image objects; 3rd group of services related to context objects (c. 5, ll. 59-60; c. 6, ll. 51-56); 4th group of services related to vector objects (c. 7, ll. 14-17).

38. As per Claim 80, McCrossin teaches 1st group of services have 1st functions to create filter objects; 2nd functions to set filter object parameters; 3rd functions to obtain filter object output (c. 2, ll. 25-32).

Final Office Action dated 21 July, 2008 at pp. 13-14.

As discussed earlier, McCrossin proposes a system for changing the format of an image by applying a series of filters. Thus, the API discussed by McCrossin does not disclose an image processing API as set forth in claim 79. Rather, McCrossin's only APIs are "read" and "write" APIs associated with "an improved method and system for transforming image data from one format to another wherein the number of converters required is reduced." McCrossin at Col. 1 lines 19-21. The Examiner cites to, "An application (not shown) calls the transform object using a procedure call or API call, which results in call 166 being made." (McCrossin at Col. 7, lines 33-35). However, as clearly seen in McCrossin's Figures 8A and 8B, element 166 is a common read function call (specifically "fread"). Therefore, McCrossin is not teaching any kind of API other than a common "read" API that performs a read function on an image whose format will be changed under McCrossin's principles. The Examiner is ***incorrect*** in inferring that McCrossin teaches the specific bundle of API services as claimed by merely stating that an API call or procedure call is used. Simply put, the disclosure in McCrossin comprises only read and write APIs and a translator of image formats. In dramatic contrast, Assignee's invention in claim 79 is for a specifically claimed "image processing application program interface;" *i.e.*, APIs to filter objects, APIs to image objects, APIs to context objects and APIs to vector objects. These two concepts are dramatically different.

The Examiner also states that McCrossin's "[t]ransform object 103 determines an actual image vector 139, which describes the format of the image to be read or written" (McCrossin at Col. 6, lines 51-53). The Examiner then states "Since this describes the format of the image, this is related to image objects and context objects." (*See* Office Action dated 30 March 2007 at ¶ 19 page 18). Again, the Examiner has made a speculative and incorrect extrapolation; simply describing the format of the image does not encompass the requirements of image objects and context objects. As will be seen in the discussion of claim 84 below, the Examiner appears to have an incorrect understanding of the "context objects" as disclosed and claimed by Assignee. For example, in the Office Action dated 30 March 2007 at ¶ 24 page 19, the Examiner misquotes Assignee's specification to say, "creating a context is derived from tools that allow the definition of an object in memory." The correct quotation from Assignee's specification is "the ***ability*** to create a context is derived from tools that allow definition of an object in memory." This simply means that Assignee is teaching that a "context" may be created using "tools that allow definition of an object in memory." The Examiner seems to have incorrectly interpreted the partial quotation to believe that any definition of an object in memory is creating a context. As shown above, this is simply not true. McCrossin does not teach or discuss context creation and the Examiner's belief to the contrary appears to be based on his belief that any definition of an object in memory is creating a context.

McCrossin does not and cannot disclose the bundle of expressly claimed API services; *i.e.,* API services related to filter objects, API services related to image objects, API services related to context objects and API services related to vector objects. Such a bundle of API services as claimed is entirely absent from McCrossin at least because a major purpose of McCrossin is to isolate the application layer from the task of doing a file format conversion. For example, the McCrossin application uses a common "read" or "write" API to obtain a file that is in a format associated with the isolated application program. McCrossin's API does NOT call any or all "filters" involved in the file format conversion because the whole purpose of McCrossin is to allow the format change without the detailed API call. The Examiner has found a single reference to "a procedure call or API call" in McCrossin at Col. 7, lines 33-35 and inappropriately extrapolated this reference to contend that McCrossin teaches the specifically claimed limitations of independent claim 79. Thus, in view of this clarification, Assignee

submits that claim 79 should be in condition for allowance. Claims 80 through 83 depend from claim 79 and therefore should be allowable for the same reasons. Assignee respectfully requests the Board withdraw this rejection.

The Examiner has rejected independent claim 84 specifically as follows:

> 42. As per Claim 84, McCrossin teaches application calls transformation object using API call (c. 7, ll. 33-35). Filters are accessed by transformation object from filter library 143. Filter library contains number of different filters available for performing various image transformations (c. 6, ll. 59-62). Transform object adds filter to filter stack to create new filter object (c. 6, ll. 51-58; c. 8, ll. 25-37). Since McCrossin teach creating new file object for filter stack (c. 8, ll. 25-37), this is considered to be creating filter object. According to Applicant's disclosure, image, more commonly, may be created from another image by applying filter to existing image ([0057], p. 12). So, McCrossin teaches API functions to create image objects; API functions to create filter objects. Transform object 103 determines actual image vector 139, which describes format of image to be read or written (c. 6, ll. 51-53). According to Applicant's disclosure, creating context is derived from tools that allow definition of object in memory ([0055], p. 11). Since actual image vector describes format of image to be read or written, this allows definition of object in memory, so is considered to create context objects, so McCrossin teaches API functions to create context objects. Pointer to filter environment in filter object 150 points to memory 156, which contains information which may be used by filter vector 154 in converting or transforming image data (c. 7, ll. 21-24), and so McCrossin does teach API functions to set filter object parameters. Filters are employed by transformation object 103 to provide necessary transformation of image data to return image data in form as specified in image request vector 127 (c. 6, ll. 66-c. 7, ll. 2), and so McCrossin teaches API functions to obtain filter object output; API functions to convert image objects to context objects. So, McCrossin teaches application program interface for facilitating image processing (c. 6, ll. 25-30), API having functions to create image objects; create context objects (c. 6, ll. 51-53); create filter objects; set filter object parameters; obtain filter object output (c. 2, ll. 25-32); convert image objects into context objects (c. 6, ll. 51-53).

Final Office Action dated 21 July, 2008 at pp. 15-16.

Similar to claim 79, the Examiner asserts that claim 84 is invalid over McCrossin. Assignee submits that claim 84 is in condition for allowance because, as explained with respect to claim 79, McCrossin discloses only read and write APIs in the context of image format translation and does not and cannot disclose the specifically claimed bundle of API functions;

i.e., APIs to create image objects; APIs to create context objects; APIs to create filter objects; APIs to set filter object parameters; APIs to obtain filter object output; and APIs to convert image objects to context objects. In particular, the Examiner cites McCrossin at Col. 6 lines 59-62, which clearly states that "Filters are accessed ... from filter library". The Examiner has found a reference in McCrossin that discloses the application of a filter to an image. However, it is not valid for the Examiner to extrapolate that reference to an API to create filter objects rather than retrieve them from a pre-existing library. In response to previous arguments, the Examiner asserts the following:

> In reply, McCrossin teaches application calls transformation object using API call (c. 7, ll. 33-35), transform object adds filter to filter stack to create new filter object (c. 6, ll. 51-58; c. 8, ll. 25-37).

Office Action dated 30 January 2008 at p. 4.

The Examiner has incorrectly paraphrased the citation because there is no teaching in the cited disclosure related to "create new filter object". The cited section of McCrossin does not create a new filter object but creates a "new file object" and filters are only pushed and popped from a ***filter stack***. *See*, for example, Figs. 6 and 7 which depict a representation of the filter stack.

In addition, as explained above for claim 79, the Examiner seems to have incorrectly interpreted the partial quotation to believe that any definition of an object in memory is creating a context. This is simply not true and not what is stated in Assignee's disclosure. McCrossin does not teach or discuss context creation and the Examiner's belief to the contrary appears to be based on his belief that any definition of an object in memory is creating a context. Thus, in view of this clarification, Assignee submits that claim 84 should be in condition for allowance. Assignee respectfully requests the Board withdraw this rejection.

C. Section 103 Rejections

The Examiner has rejected claims 1-2, 8, 11-12, 14-20, 36-37, 40-41, 51-52, 55-56, 58, 61-63, 66-73 and 85 under 35 U.S.C. § 103(a) as allegedly being obvious over Grantham in view of McCrossin.

The Examiner has specifically rejected independent claim 1 as follows:

47. As per Claim 1, Grantham teaches processor 108 initiates request which is routed to server 101. When server 101 receives such request, processor 102 retrieves appropriate VRML file. VRML file instructs API 112 to make a number of function calls to various graphics engines 113 for performing desired functions on persistent data objects 118. Interpreter 111 modifies scene graph to suit specific graphics subsystem hardware 109. Hence, interpreter 111 enables VRML file to be adapted to run on graphics subsystem hardware (c. 4, ll. 63-c. 5, ll. 11). API is structured as collection of class hierarchies. Nodes and graphics states are assembled into cohesive scene graph (c. 5, ll. 18-23). Graphics State classes include Texture class 504 that defines how image is filtered (c. 28, ll. 10-16; c. 11, ll. 22-25). Outside Scene Graph classes include CompileAction class 403 that compiles specified subgraph into data structure which is more efficient for traversals (c. 28, ll. 25-27, 29-30; c. 7, ll. 50-57). So filter is part of scene graph (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25) that is run on graphics subsystem hardware (c. 5, ll. 5-11). Before scene graph is run on graphics subsystem hardware (c. 5, ll. 5-11), interpreter 111 modifies scene graph to suit specific graphics subsystem hardware 109. Interpreter 111 enables VRML file to be adapted to run on virtually any type of machine (c. 5, ll. 5-9). So interpreter 111 translates VRML file to another computer language that is able to be run on graphics subsystem hardware 109, and so interpreter 111 is considered to perform compiling, and compiled program includes filter. So VRML file requested by processor 108 (c. 4, ll. 63-c. 5, ll. 11) includes filter (c. 5, ll. 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25), and processor 108 requests filter from compiling process since program needs to be compiled before it is run on graphics system hardware. So Grantham teaches method of editing initial image, having steps of 1st process requesting (c. 4, ll. 63-c. 5, ll. 11) filter from 2nd process; related filter and initial image comprising program, 2nd process compiling program, yielding compiled program; running at least portion of compiled program to apply function of filter to image (c. 5, ll. 5-11, 18-23; c. 28, ll. 10-16; c. 11, ll. 22-25; c. 7, ll. 50-57), yielding pixel-image result (c. 5, ll. 9-11).

However, Grantham does not teach 1st process defines relationship between filter and initial image. But, McCrossin teaches editing initial image, 1st process requesting filter from 2nd process; 1st process defining relationship between filter and initial image (c. 6, ll. 46-c. 7, ll. 9).

It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham so first process defines relationship between filter and initial image because McCrossin suggests reducing amount of processing required to convert image data from original or input format into desired output format (c. 1, ll. 54-57; c. 2, ll. 3-24).

Final Office Action dated 21 July, 2008 at pp. 16-18.

As stated above, the CompileAction class of Grantham is not really "compiling" and does not compile in the manner of Assignee's claim because it merely produces a data structure which is easier to traverse; so it is not a compiled program as recited in claim 1. Additionally,

CompileAction is a ***class*** (as in "class" in object oriented programming construct) and not ***a separate process*** (or a process at all) as recited in claim 1.

Furthermore, the Examiner has again extrapolated Grantham unreasonably because, as stated above, Grantham is directed to requesting VRML files over the internet and has nothing to do with the claimed multi-process method of applying filter's to produce a pixel-image result. The Examiner merely synthesizes his conclusion that a "filter is part of scene graph." Final Office Action dated 21 July, 2008 at pg. 17. The VRML file of Grantham is not a graphics item and is a much higher level collection of information than a graphics filter. Because the VRML file is a high level file and can be interpreted into something applicable to any type of machine, there is no reason to even infer that the VRML file will contain the claimed graphics processing filter. Therefore, Assignee contends that the conclusion made by the Examiner is improper and clearly is not supported by the cited reference.

In response to the previous arguments, the Examiner states "Grantham is used to teach multiple processes and specific division of work between multiple processes. McCrossin is merely used to teach 1st process defines relationship between filter and initial image." Final Office Action dated 21 July, 2008 at pg. 8. As stated above, Grantham in no way requests a filter from a second process because Grantham requests a VRML file. Furthermore, the plain language of claim 1 discloses that the ***same process*** that requests a filter from a second process is the process that defines a relationship between the filter and the image. The combination of Grantham and McCrossin cannot logically disclose this limitation because, as admitted by the Examiner, McCrossin does not teach multiple processes and Grantham does not disclose a defined relationship.

Thus, in view of this clarification, Assignee submits that claim 1 should be in condition for allowance. Assignee respectfully requests the Board withdraw this rejection.

The Examiner has specifically rejected independent claim 58 as follows:

> 62. As per Claim 58, Grantham teaches interpreter 111 enables VRML file to be adapted to run on graphics system hardware 109 (c. 5, ll. 5-9). VRML is high level programming language (c. 1, ll. 63-67). So Grantham teaches providing high level interface (111) to graphics processing resource (109) (c. 5, ll. 5-9; c. 1, ll. 63-67) having 1st process requesting performance of task through 1 or more function calls serviced by graphics processing resource (c. 4, ll. 63-c. 5, ll. 23), function calls selected from 1 or more of following options, (i) creating of image (Graphics State classes) (c. 28, ll. 14-16; c. 8, ll. 37-46; c. 7, ll. 31-33), (iv) asking for output of filter or another filter (504; c. 11, ll. 22-25), (v) creating context (302); (vi) rendering image to context or another context; request having object associated therewith, object comprising one of following: context (c. 5, ll. 61-67), image (c. 28, ll. 14-16; c. 8, ll. 37-46; c. 7, ll. 31-33), filter (c. 11, ll. 22-25), or vector (c. 26, ll. 20-22); graphics processing resource servicing request (c. 5, ll. 2-23).
>
> However, Grantham does not explicitly teach request defines relationship between at least one of functions and one of objects. However, McCrossin teaches this limitation (c. 6, ll. 46-c. 7, ll. 9). This would be obvious for the same reasons given in the rejection for Claim 1.

Final Office Action dated 21 July, 2008 at pp. 20-21.

The Examiner has not provided a reference for either limitation of "(ii) creating a filter" or "(iii) setting arguments associated with said filter" and both McCrossin and Grantham are inadequate, as described above. Thus, in view of this clarification, Assignee submits that claim 58 should be in condition for allowance. Assignee respectfully requests the Board withdraw this rejection.

The Examiner has specifically rejected independent claim 63 for the same rationale as claim 58 discussed above. The arguments regarding claim 58 apply with equal force to claim 63. Thus, in view of this clarification, Assignee submits that claim 63 should be in condition for allowance. Assignee respectfully requests the Board withdraw this rejection.

The Examiner has specifically rejected independent claim 85 as follows:

> 72. As per Claim 85, Grantham teaches computer executable instructions for performing method recited in Claim 1 (c. 4, ll. 63-c.5, ll. 23).

Final Office Action dated 21 July, 2008 at p. 21.

Because claim 85 is a media claim based upon the method of claim 1, the arguments regarding claim 1 apply with equal force to independent claim 85. Thus, in view of this clarification, Assignee submits that claim 85 should be in condition for allowance. Assignee respectfully requests the Board withdraw this rejection.

The Examiner has rejected claims 22-23 and 25-27 under U.S.C. § 103(a) as allegedly being obvious over Grantham and McCrossin further in view of U.S. Patent No. 6,600,840 to Hoppe ("Hoppe").

The Examiner has specifically rejected independent claim 22 as follows:

> 87. As per Claim 22, Grantham teaches editing initial image, 1st microprocessor (108, 111, 112, Fig. 1) running 1st process and 2nd process (compiling) for servicing requests from first process (c. 4, ll. 63-c. 5, ll. 11; c. 7, ll. 50-57). API is structured as collection of class hierarchies (c. 5, ll. 16-23) that include Texture class 504 that defines how image is filtered (c. 11, ll. 22-25). So filter is in API (c. 5, ll. 16-23; c. 11, ll. 22-25), and API 112 is stored in memory 110, as shown in Fig. 1. So memory 110 is for storing filter, 2nd memory for storing data structure comprising information used in 1st process (c. 5, ll. 16-23; c. 11, ll. 22-25; Fig. 1) (1st and 2nd memories are same, as recited in Claim 23); 2nd microprocessor 109 for running program created using data structure; outputting pixel image resulting from running program (c. 5, ll. 2-11).
>
> But, Grantham doesn't teach data structure has relationship between initial image and filter. But, McCrossin teaches data structure 103 has relationship between initial image and specific filter (*image request vector 127 is received from application 101 and is compared to actual image vector to determine what filters are needed*, c. 6, ll. 46-c. 7, ll. 9; *application calls the transformation object using API call*, c. 7, ll. 33-35). Fig. 6 shows transform object 103 (data structure) comprises image request vector 127 that is compared to actual image vector to determine what filters are needed (c. 6, ll. 46-c. 7, ll. 9). Since transform object 103 compares actual image vector to determine what filters are needed, this means that transform object 103 determines from image what filters are needed, and so transform object 103 (data structure) comprises relationship between image and filter. This is obvious for reasons for Claim 1.
>
> But, Grantham, McCrossin don't teach 3rd memory for storing pixel image resulting from running program. But, Hoppe teaches API (c. 2, ll. 20-26) for filtering (c. 1, ll. 34-36), 3rd memory 208 for storing pixel image resulting from running program (c. 4, ll. 36-37; c. 9, ll. 33-36; c. 1, ll. 55-58).
>
> It would have been obvious to one of ordinary skill in the art at the time of invention by applicant to modify Grantham and McCrossin to include 3rd memory for storing pixel image resulting from running program because Hoppe suggests it is well-known in the art to have frame buffer in video memory to store rendered image so that rendered image can be output from video memory to display (c. 4, ll. 36-37; c. 9, ll. 33-36).

Final Office Action dated 21 July, 2008 at pp. 25-26.

The Examiner asserts that through disclosure of the "CompileAction" class, Grantham teaches "a second process (compiling)." As discussed above, the CompileAction class disclosed in Grantham is simply a "class" *i.e,* an object oriented programming construct. Grantham is

silent as to multiple processes running on a first microprocessor because it is directed to sending control information across the internet so that a web browser can show a 3D image.

The Examiner also asserts in rejecting claim 22 that McCrossin teaches data structure 103 has the claimed relationship. In particular, the Examiner states:

> In reply, Examiner points out McCrossin teaches transform object 103 has relationship between initial image and specific filter (c. 6, ll. 51-56), so transform object 103 is data structure.

Office Action dated 30 January 2008 at p. 3.

Throughout this and other Office Actions, the Examiner has interpreted the transform object 103 as an API or subroutine. *See* for example rejections to claim 79 and 84. Now, in rejecting claim 22 the Examiner asserts that transform object 103 defines a relationship and discloses the ***data structure*** of independent claim 22. This is simply an improper interpretation of McCrossin and an unfair use of a varying "definition." McCrossin's transform object 103 does not teach a data structure comprising a relationship between an image and a filter.

Thus, in view of this clarification, Assignee submits that claim 22 should be in condition for allowance. Furthermore, claims 23 and 25-27 depend from claim 22 and should be patentable for at least the same reasons. Assignee respectfully requests the Board withdraw this rejection.

The Examiner has rejected claims 3, 4-7, 9-10, 13, 21, 24, 30-35, 42, 45-50, 57, 59-60, 64-65 and 76-77 under U.S.C. § 103(a) as allegedly being obvious over Grantham and various other references. Each of these claims is a dependent claim and is patentable at least for the reasons already stated regarding its' parent independent claim. Assignee respectfully requests the Board withdraw this rejection.

D. Conclusion

Assignee has addressed all of the Examiner's rejections and has shown a reason for reversal of each of these rejections. The Examiner has combined many different references to contend that they disclose Assignee's claims. This is simply not the case. Each of these references, even when taken together, do not disclose each and every limitation of Assignee's claims. In fact, the references are taken out of context and then unreasonably combined with

other non-applicable references. The cited prior art simply cannot support a rejection based on anticipation or obviousness.

In summary, Assignee traverses all of the Examiner's art-based rejections because the primary references used by the Examiner are plainly inapplicable to the inventions claimed by Assignee. Furthermore, in addition to alleging the application of distant and literally inapplicable art, the Examiner's specific ***allegations*** regarding each claim limitation do not incorporate the claimed connectivity between claim elements. For example, with respect to Claim 22, the Examiner has alleged McCrossin teaches that a data structure comprises a relationship between an image and a filter. While the Examiner's cited excerpt (Col 6, line 46 – Col. 7, line 9) does discuss "***filters***" and "***images***," it most certainly does NOT discuss any ***data structure comprising a relationship between an initial image and a specific filter.*** The Examiner's rejections must fail for lack of allegations regarding Asssignee's claimed connectivity between elements.

With regard to all the Section 103 rejections, in addition to misapplying the art, the Examiner has failed to show any motivation to combine references. For example, the Examiner's primary references are Grantham, Boudier and McCrossin. The Examiner's primary reference of Grantham relates to system for requesting VRML files to "implement a retained mode graphics" for rendering a 3D scene according to a downloaded file. This reference is obscure to the mainstream of graphics processing and image processing, thus non-analogous in every sense. The Examiner also relies heavily on the McCrossin reference, which relates to using the sequential application filters to change the file format of an image (*e.g.*, from BMP to JPG, etc.). A skilled artisan working in the field of the Assignee's claims would not likely be inclined to find even one of these references. But, even if one were found (accidentally), since the topic of the other is not analogous to anything else, there would be no motivation to find or combine the two. This is particularly so for the purpose of developing the claimed invention.

Assignee submits that the cited references alone or in any combination fail to teach each recited element in the rejected claims. Accordingly, Assignee respectfully requests the Board reverse the Examiner's rejections.

Respectfully submitted,

/William M. Hubbard/
William M. Hubbard, J.D.
Reg. No. 58,935

Wong, Cabello, Lutsch,
Rutherford & Brucculeri, L.L.P.

Customer No. 61947
20333 State Highway 249, Suite 600
Houston, Texas 77070

Voice: 832-446-2445
Facsimile: 832-446-2424
Email: wcpatent@counselIP.com

VIII. CLAIMS APPENDIX

We claim:

1. (Previously presented) A method of editing an initial image, comprising the steps of:
   - a first process requesting a filter from a second process;
   - said first process defining a relationship between said filter and said initial image, said related filter and initial image comprising a program,
   - said second process compiling said program, yielding a compiled program;
   - running at least a portion of said compiled program to apply a function of said filter to said image, yielding an pixel-image result.

2. (Previously presented) The method of claim 1 having the additional step of
   - optimizing said program.

3. (Original) The method of claim 2, wherein the step of optimizing includes the step of using a cache look-up to see if said pixel-image result is already in cache.

4. (Original) The method of claim 2, wherein the step of optimizing includes the step of calculating an intersection, said intersection representing an area where said pixel-image result is both defined and in a result region requested of said second process.

5. (Previously Presented) The method of claim 4 further comprising the step of using said calculated intersection to limit the number of pixels that require calculation during said running of said compiled program.

6. (Original) The method of claim 4 further comprising the step of, using said calculated intersection to limit the amount of memory necessary for storing calculated images.

7. (Original) The method of claim 1 wherein said compiled program is for a single microprocessor.

8. (Original) The method of claim 1 wherein said compiled program comprises a component for a first microprocessor and a component for a second microprocessor.

9. (Original) The method of claim 7 wherein said single microprocessor is a CPU.

10. (Original) The method of claim 7 wherein said single microprocessor is a programmable GPU.

11. (Original) The method of claim 8 wherein said first microprocessor is a CPU and said second microprocessor is a GPU.

12. (Original) The method of claim 8 wherein said first and second microprocessors are both CPUs.

13. (Original) The method of claim 8 wherein said first and second microprocessors are both GPUs.

14. (Original) The method of claim 1 wherein said initial image is only a color.

15. (Original) The method of claim 1 wherein said first process is an application program.

16. (Original) The method of claim 1 wherein said second process comprises a suite of graphics services functions.

17. (Original) The method of claim 1 wherein an operating system comprises said second process.

18. (Original) The method of claim 1 wherein said first process requests a high-level filter and said second process responds with an object representing a lower-level filter.

19. (Original) The method of claim 1 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.

20. (Original) The method of claim 2 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.

21. (Original) The method of claim 5 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.

22. (Previously presented) A system for editing an initial image, comprising:
- a first microprocessor running a first process and a second process for servicing requests from said first process;
- a memory for storing a filter, said filter requested by said first process;
- a second memory for storing a data structure comprising a relationship between said initial image and said filter, said first process causing the creation of said data structure;
- a second microprocessor for running a program created using said data structure;
- a third memory for storing a pixel image resulting from running said program.

23. (Original) The system of claim 22 wherein said first and second memories are the same.

24. (Previously Presented) The system of claim 22 wherein said first, second and third memories are the same.

25. (Original) The system of claim 22 wherein said first and second memories are in system memory and said third memory is in video memory.

26. (Original) The system of claim 22 wherein said first microprocessor processes said data structure to produce said program for use on said second microprocessor.

27. (Original) The system of claim 22 wherein said second microprocessor, under control of said program, causes said pixel image to be stored in said third memory.

28. (Previously Presented) A method of creating a result image comprising the steps of:
- a first process requesting the creation of a context;
- said first process requesting the creation of a result image;
- said first process indicating parameters associated with the creation of said result image;
- said first process requesting that said result image be rendered to said context;
- a second process servicing said requests of said first process, said servicing comprising the steps of:
  - optimizing a graph representing said result image;
  - compiling said optimized graph;
  - causing the rendering of said compiled optimized graph into said context.

29. (Original) The method of claim 28 wherein said creation of said result image comprises editing a pre-existing image.

30. (Previously presented) The method of claim 28 wherein said step of optimizing a graph representing said first image comprises the step of:
- calculating an intersection, said intersection representing an area where said result image is both defined and requested as a result by said first process.

31. (Previously presented) The method of claim 28 wherein said step of optimizing a graph representing said first image comprises the step of:
- analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

32. (Previously presented) The method of claim 30 wherein said step of optimizing a graph representing said first image comprises the additional step of:
- analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

33. (Original) The method of claim 31 wherein the step of analyzing adjacent nodes comprises the step of checking a cache to determine if the result of such analysis is available in a memory.

34. (Original) The method of claim 28 wherein the step of optimizing said graph comprises the step of checking a cache to determine if said graph has already been optimized.

35. (Original) The method of claim 28 wherein the step of optimizing said graph comprises the step of checking a cache to determine if the result of rendering said graph is available in a memory.

36. (Original) The method of claim 28 wherein said first process requests the output of a graph programmatically assembled by said first process and comprising one or more pre-defined filters.

37. (Original) The method of claim 36 wherein said first process programmatically assembled said graph in cooperation with said second process.

38. (Original) The method of claim 28 wherein said first process is an application program.

39. (Original) The method of claim 28 wherein said second process comprises a suite of graphics services functions.

40. (Original) The method of claim 39 wherein an operating system comprises said second process.

41. (Original) The method of claim 28 wherein an operating system comprises said second process.

42. (Original) The method of claim 28 wherein said second process inserts nodes into said graph for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme.

43. (Original) A method of creating an result image comprising the steps of:
- a first process running on a first microprocessor requesting the creation of a context;
- said first process requesting the creation of a result image;
- said first process indicating parameters associated with the creation of said result image;
- said first process requesting that said result image be rendered to said context;
- a second process running on said first microprocessor servicing said requests of said first process, said servicing comprising the steps of:
  - optimizing a graph representing said result image;
  - compiling said optimized graph;
  - causing rendering of said compiled optimized graph into said context, said rendering occurring on a second microprocessor.

44. (Original) The method of claim 43 wherein said creation of said result image comprises editing a pre-existing image.

45. (Previously presented) The method of claim 43 wherein said step of optimizing a graph representing said first image comprises the step of:
- calculating an intersection, said intersection representing an area where said result image result is both defined and requested by said first process.

46. (Previously presented) The method of claim 43 wherein said step of optimizing a graph representing said first image comprises the step of:
- analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

47. (Previously presented) The method of claim 45 wherein said step of optimizing a graph representing said first image comprises the additional step of:
- analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

48. (Original) The method of claim 46 wherein the step of analyzing adjacent nodes comprises the step of
- checking a cache to determine if the result of such analysis is available in a memory.

49. (Original) The method of claim 43 wherein the step of optimizing said graph comprises the step of
- checking a cache to determine if said graph has already been optimized.

50. (Original) The method of claim 43 wherein the step of optimizing said graph comprises the step of
- checking a cache to determine if the result of rendering said graph is available in a memory.

51. (Original) The method of claim 43 wherein said first process requests the output of a graph programmatically assembled by said first process and comprising one or more pre-defined filters.

52. (Original) The method of claim 51 wherein said first process programmatically assembled said graph in cooperation with said second process.

53. (Original) The method of claim 43 wherein said first process is an application program.

54. (Original) The method of claim 43 wherein said second process comprises a suite of graphics services functions.

55. (Original) The method of claim 54 wherein an operating system comprises said second process.

56. (Original) The method of claim 43 wherein an operating system comprises said second process.

57. (Original) The method of claim 43 wherein said second process inserts nodes into said graph for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme.

58. (Previously presented) A method for providing a high level interface to a graphics processing resource comprising:
- a first process requesting performance of a task through one or more function calls serviced by said graphics processing resource, said function calls selected from one or more of the following options, (i) creating of an image, (ii) creating a filter, (iii) setting arguments associated with said filter, (iv) asking for the output of said filter or another filter, (v) creating a context; and (vi) rendering an image to said context or another context;
- said request having an object associated therewith, said object comprising one of the following: a context, an image, a filter, or a vector;
- said request defining a relationship between at least one of said functions and one of said objects,
- said graphics processing resource servicing said request.

59. (Original) The method of claim 58 wherein said request is serviced using a GPU and a CPU.

60. (Original) The method of claim 58 wherein said request is serviced using an emulator to run a GPU program on a CPU.

61. (Original) The method of claim 58 comprising the additional step of creating a graph representing an image.

62. (Original) The method of claim 61 comprising the additional step of optimizing said graph.

63. (Previously presented) A method for providing a high level interface to a graphics processing resource comprising:
- a first process requesting performance of a task through one or more function calls serviced by said graphics processing resource, said function calls selected from one or more of the following options, (i) creating of an image, (ii) creating a context; and (v) rendering an image to said context or another context;
- said request having an object associated therewith, said object comprising one of the following: a context, or an image
- said request defining a relationship between at least one of said functions and one of said objects,
- said graphics processing resource servicing said request.

64. (Original) The method of claim 63 wherein said request is serviced using a GPU and a CPU.

65. (Original) The method of claim 63 wherein said request is serviced using an emulator to run a GPU program on a CPU.

66. (Original) The method of claim 63 comprising the additional step of creating a graph representing an image.

67. (Original) The method of claim 66 further comprising the step of optimizing said graph.

68. (Original) The method of claim 63, wherein said function calls include the option of creating a filter.

69. (Original) The method of claim 63 wherein said function calls include the option of setting arguments associated with said filter.

70. (Original) The method of claim 68 wherein said function calls include the option of setting arguments associated with said filter

71. (Previously Presented) The method of claim 63 wherein said another object associated with said request may comprise a filter.

72. (Original) The method of claim 63 wherein another object associated with said request may be a vector.

73. (Original) The method of claim 71 wherein another object associated with said request may be a vector.

74. (Previously presented) A method for rendering an image comprising the steps of:
- a first process running on a CPU requesting the creation of an image;
- a graphics services resource, responding to said request by running a first routine on said CPU, said first routine for optimizing a graph representing said image;
- said first process requesting the rendering of said image to a specified context;
- said graphics services resource causing the rendering of said graph representing said image, said rendering occurring on a GPU.

75. (Previously presented) A method for converting a first image representation into a second image representation, comprising the steps of:
- creating a first graph associated with said first image representation where software routines for creating such graph execute on a CPU,
- determining an intersection of the first graph's global domain of definition and global region of interest;
- resolving a first node in said first graph by running software routines on said CPU to (i) determine if said first node may be combined with a second node and (ii) create program steps for calculating and storing only the portions of any intermediary image that relate to said intersection, said program steps for compilation on said CPU and execution on a GPU.

76. (Original) The method of claim 75 wherein the step of determining if said first node may be combined with said second node comprises the step of checking a cache to determine if there is a result in memory regarding such determination regarding combining nodes.

77. (Original) The method of claim 75 wherein the step of resolving said first node comprises the step of checking a cache to determine if there is a result in memory regarding the resolution of said first node.

78. (Original) The method of claim 75 wherein said first node is a root node.

79. (Original) An image processing application program interface embodied on one or more computer readable media, comprising: a first group of services related to filter objects; a second group of services related to image objects; a third group of services related to context objects; and a fourth group of services related to vector objects.

80. (Original) The image processing application program interface of claim 79, wherein the first group of services comprise: first functions to create filter objects; second functions to set filter object parameters; and third functions to obtain filter object output.

81. (Original) The image processing application program interface of claim 79 wherein the second group of services comprise: first functions to create image objects; and second functions to render an image object to a context object.

82. (Original) The image processing application program interface of claim 79, wherein the third group of services comprise first functions to create context objects.

83. (Original) The image processing application program interface of claim 79 wherein the fourth group of services comprise: first functions to create vector objects.

84. (Original) An application program interface for facilitating image processing, the application program interface comprising functions to: create image objects; create context objects; create filter objects; set filter object parameters; obtain filter object output; and convert image objects into context objects.

85. (Original) A computer-readable medium having computer executable instructions for performing the method recited in any one of claims 1, 28, 43, 58, 63, 74 or 75.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.

TRANSMITTAL OF APPEAL BRIEF (Large Entity)	Docket No. 119-0033US

In Re Application Of: **John HARPER, Ralph BRUNNER, Peter GRAFFAGNINO, Mark ZIMMER**

Application No.	Filing Date	Examiner	Customer No.	Group Art Unit	Confirmation No.
10/826,762	April 16, 2004	Joni Hsu	61947	2628	1223

Invention: **High-Level Program Interface for Graphics Operations**

COMMISSIONER FOR PATENTS:

Transmitted herewith is the Appeal Brief in this application, with respect to the Notice of Appeal filed on:

October 21, 2008

The fee for filing this Appeal Brief is: **$40.00**

☐ A check in the amount of the fee is enclosed.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. 501922. I have enclosed a duplicate copy of this sheet.

☐ Payment by credit card. Form PTO-2038 is attached.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

/William M. Hubbard/
Signature

Dated: **December 22, 2008**

William M. Hubbard, Reg. No. 58,935
Wong, Cabello, Lutsch, Rutherford & Bruccueleri, LLP
20333 State Highway 249, Suite 600
Houston, Texas 77070
wcpatent@counselip.com
832/446-2400
832/446-2424 (facsimile)

cc:

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" [37 CFR 1.8(a)] on

December 22, 2008
(Date)

via USPTO EFS by: /Rebecca R. Ginn/
Signature of Person Mailing Correspondence

Rebecca R. Ginn
Typed or Printed Name of Person Mailing Correspondence